Dynamic Modeling of Distributed Product Development Processes

Jun Lin[1], Kah Hin Chai[2], Yoke San Wong[3], Aarnout C. Brombacher[4]

[1]Design Technology Institute, National University of Singapore, Singapore
linjun@nus.edu.sg

[2]Department of Industrial and Systems Engineering, National University of Singapore, Singapore
iseckh@nus.edu.sg

[3]Department of Mechanical Engineering, National University of Singapore, Singapore
mpewys@nus.edu.sg

[4]Faculty of Technology Management, Eindhoven University of Technology, The Netherlands
A.C.Brombacher@tm.tue.nl

Abstract
Market and technology changes have brought about new characteristics of product development. One of the most significant changes from the traditional to the new paradigm is the change from sequential and collocated development processes to concurrent and distributed processes. Although some researchers have built models of development processes and product development performance, most of these studies are about collocated development projects where the information flows between development teams is not explicitly studied. Consequently, there is a need to model the relationships between development processes and project cycle time in the distributed context, with special attention to the information flows between development teams. With the support of a design company, we developed and validated the model with data from mobile phone projects.

Introduction

The growth of distributed development in general in the past few years has been facilitated due to advances made in the computing world and particularly the creation and growth of the Internet. The present industrial climate also results in the growth of distributed design. Continual technological advancement is set against a backdrop of international partnerships which invariably leads to more distributed collaboration in new product development. Further, recent movements and concepts within the academic and industrial world such as distributed design, collaborative product development, outsourcing and the concept of the extended enterprise, promote the growth of distributed development. The realization that collaborators external to the organization have higher levels of expertise continues to drive this increase. We list some examples as following:

- Chrysler no longer writes detailed specifications for many parts. Instead, it relies on suppliers to design and build the right parts and to find ways to lower prices. Chrysler and the supplier split the savings (Minahan, 1998).
- Apple hired Sony to design the structure of the PowerBook because of its specialization in miniaturization. As a result the size and development time was reduced (Magee, 1992).
- Boeing, Whirlpool, and McDonnell Douglas have outsourced many of their design activities to other firms (Proctor, 1999).
- 70% of engineers' time is spent on activities concerning distant suppliers within one European Original Equipment Manufacturer (OEM) in the automotive industry (Siemieniuch and Sinclair, 1999).
- Zhao (2003) studied six U.S. and Singapore firms selected from electronics, personal computer, heavy machinery and steel industries. A trend of NPD outsourcing was found in these firms. In the 1980s, outsourcing percentage is 17%-36%. In 2000, outsourcing percentage increases to 47%-70%.
- Designs outsourced in PDAs, notebook PCs and mobile phones are 70%, 65% and 20% respectively (Engardio and Einhorn, 2005).

Although the growth of the internet has facilitated product development in a distributed sense – the development of computer tools has increased the level of communication and collaborative product development at distance, insufficient information is still a big problem in distributed development processes (McDonough III et al., 2001; Sosa et al., 2002). Lu (2002) showed a case that one piece of information (about the different methods used to test the problems between a company and its customer) was available to one location but not available to other locations where it was most needed. Thus although a lot of tests were done in the business unit and by the customer, there were still many quality problems were reported by the customer. Consequently, the development cycle time is much higher in a distributed environment than in a conventional project management environment.

Some previous research has focused on distributed or collaborative product development processes (McDonough III et al. 2001; Sosa et al. 2002; Doz and Hamel 1998) and the benefits of improving information flows between companies and teams are well understood. The Maturity Index on Reliability method is developed to classify the information flows with respect to their ability to measure, understand and improve the quality and reliability of a product (Brombacher, 1999; Sander and Brombacher, 1999). However companies still hesitate to do it because the cost of information flow improvement is high and there is no quantitative model been developed to evaluate the benefits of it.

Traditional project management models based on the Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) describe process in a static fashion with activity duration estimates and precedence relationships describing the network of development activities. Other approaches include identifying certain dynamic consequences of different project structures on project performance. For example, the dynamic consequences of iteration among project phases on project cycle time have been addressed directly with the Design Structure Matrix (Smith and Eppinger 1997; Eppinger et al. 1994; Steward 1981). The impact of overlapping and functional interaction on development time and effort is discussed based on stochastic simulation model (Bhuiyan et al. 2004). Several system dynamics project models (e.g., Williams et al. 1995; Ford and Sterman 1998) have been built on phased network structure of projects. However all of these models are focusing on the task flows, the information flows between distributed development teams which underlie task flows and drive project cycle time are not explicitly modeled and studied. This paper tries to develop a simulation model which can be used to evaluate the benefits of information flow improvement. The model is successfully validated by using data collected from several mobile phone development projects.

The Product Development Process Model

This section begins with a general overview of the distributed product development process model, follows by details of initial development processes and rework processes.

Overview of the Model

The purpose of our model is similar to the purpose of Critical Path and PERT methods: to describe the dependencies of development tasks on each other. However our model can describe these relationships in greater detail and richness than the precedence relations used in many Critical Path and PERT methods, and the information flows among development teams are explicitly modeled:

- Our model describes the dependency among tasks along the entire duration instead of only at the start and finish of the tasks as in the Critical Path and PERT methods.
- The degree of possible concurrence among tasks, the rate of development activities, and the rework probability can be changed as information available is changed.
- Our model allows the development process to vary over the life of the project depending on the frequency and type of quality problems found in different tasks. In contrast, the precedence relationships used in many Critical Path and PERT methods are static.

Our model, which is essential for us to predict the product development performance, simulates distributed product development processes. We describe product development processes with two parts: initial development processes and rework processes. They are represented by generic structures which can be used to represent product development projects with any number of development teams, and also can be used to represent sequential, concurrent, partial concurrent and iterative product development projects. In this report, we only use project cycle-time to measure product development performance. According to Ford and Sterman (1998), development processes, resources, project targets and project scope influence product development activities. Different models are needed to understand the behavior of product development and these models interact with each other. A process model simulates the constraints of the project process due to the interactions among tasks. A resource model simulates the influence of the work force and facilities available, the efficiency of team members, and the allocation of these resources. A target model simulates the modification of time, quality, and cost objectives in the project development processes in response to overall project targets and task-level performance. A scope model simulates the project scope and task level scope changes according to project performance. In our model we only focus on dynamic product development processes and project cycle time. We assume that the project scope, target, and resources are fixed at the beginning of the project and can be changed according to the simulation result to improve product development performance.

Initial development processes describe the ideal product development processes where all quality problems are discovered and solved when the task is completed. The fundamental units that flow through a project are "development activities", such as designing a keypad for a mobile phone and writing a sub-program for software. The information flows from upstream tasks to downstream task affect the development activities, so we describe the initial development processes with development activities and feed forward information flows (Table 1).

Table 1 Development Processes and Flows

Development Processes	**Flow Components**
Initial Development Processes	Development Activities and Feed Forward Information Flows
Rework Processes	Rework Activities and Feedback Information Flows

Product development, even for derivative products, is an innovative process. Consequently, many unanticipated quality problems happen during the development process. Therefore, rework processes due to quality problems is a particularly important part of our model. In our model, rework processes are composed of feedback information flows and rework activities (Table 1). The links shown in Figure 1 represent several forms of inter-task interaction. Rework activities describe the rework processes caused by quality problems. Feedback information flows denote that when quality problems caused by upstream tasks are discovered by a downstream task, the related information for rework will transfer to relevant upstream tasks. In order to solve the quality problems, the team that discovers quality problems need to cooperate with the teams that generate the problems. Good coordination between teams can reduce quality problems and increase efficiency of feedback information flows.

In a product development, not only are the number and severity of quality problems important, but also the time when these quality problems are found. For example, certain quality problems of components can only be found after the mobile phones have been assembled and tested. In order to correct the quality problems, the components have to be redesigned, and reproduced. Then the mobile phones are assembled and tested again. Quality problems in a project can increase cycle time and costs of the whole project, and can decrease the project quality. This is the main reason why we model rework processes especially.

Most system dynamics models do not distinguish the completing rate of a task for rework from the completing rate for initial development. However, from our experience, the completing rate changes (usually decreases) dynamically in the product development processes. For instance, we design a component of a mobile phone, and then we select a supplier and discuss the prices, materials and delivery time with him. Later we find that the design of this component is wrong, so we change the design and ask the supplier to produce a new one. In this rework process the time spent on selecting a supplier should be saved, and the time needed to have a contract with the supplier should be shorter.

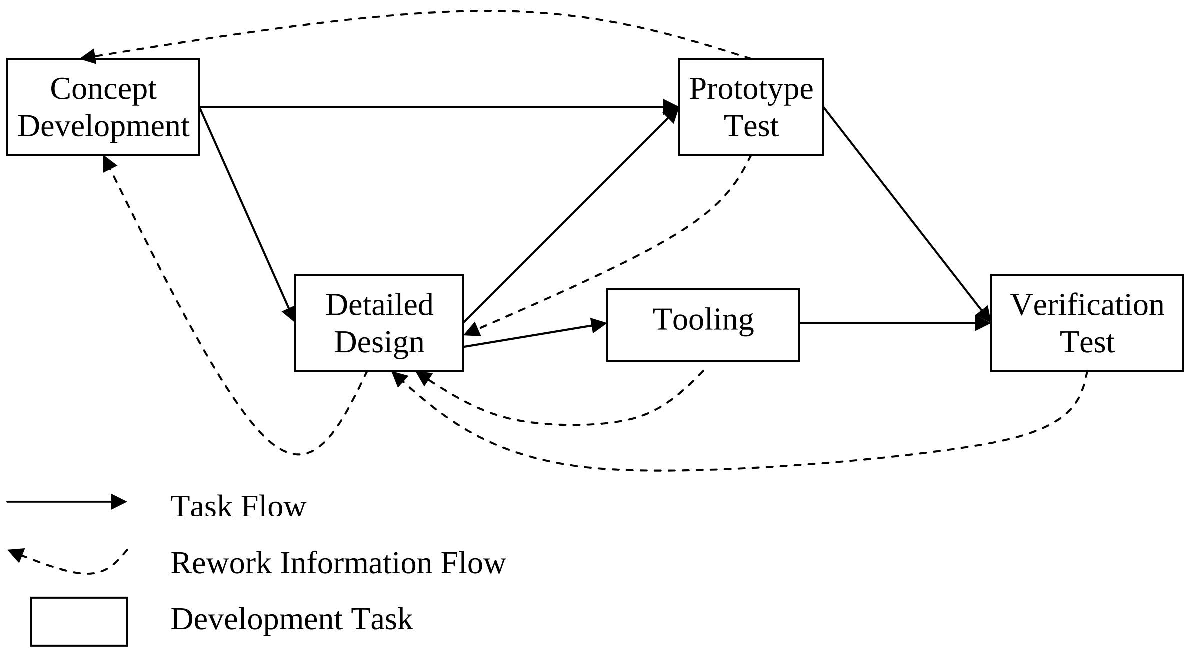

Figure 1: A Project Network

Initial Development Process

The initial development process describes what the development process is if all quality problems are discovered and solved in the task. As mentioned above, both the initial development process and rework process include task flows. Task flows related to rework processes will be described in the next section.

There are two types of tasks in a project: unconstrained and constrained. Unconstrained tasks mean that the completion of these tasks does not depend on any upstream tasks. All the information needed for these tasks will be available and developers can start to do these tasks at any time. However, most tasks in product development project are constrained tasks. These tasks have upstream tasks which constrain their completion. The information from upstream tasks will affect the development process of constrained tasks. The difference in the modeling process for unconstrained and constrained tasks will be mentioned later. Our model uses two parts to describe initial development processes: task flows, and process concurrence relationships.

Task Flows

Tasks of development processes are described in a stock and flow structure (Figure 2). We use T_n to represent the N_{th} task of the development project. In our model, a task flow includes two states: task uncompleted (TU) and task completed (TC). A task initially resides in the TU stock. The development activity in the flow is named as completing task. Completing task rate (CTr) equals to a percentage of a task completed at every step. Completed tasks accumulate in the TC stock.

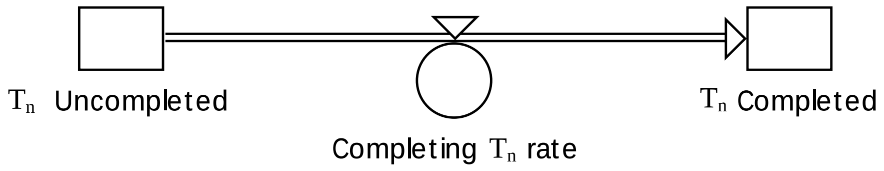

Figure 2: Stock and flow model of a single task

Completing task rate is determined by average completion duration (ACD), effects of process concurrence relationship on completing task rate (EPCRCT) and task available for completion (TAC) (Figure 3). ACD is the time required to complete a development activity on a task if the required resources and all the information needed from upstream tasks are available. It describes the basic time constraint that the process imposes on the project progress. In our model, ACD includes the time incurred by rework activities to solve any quality problems found in the self-checking process. We assume that developers will solve these quality problems immediately and will not report them. We treat the rework activities arising from these quality problems as part of the completion task flow. Completing task activity requires sufficient skilled workers as well as enough materials. Therefore, ACD for each activity is affected by the resources available. We assume that the resources for each activity in a company will remain the same. Hence, based on the aforementioned assumptions, ACD is a constant in our model.

EPCRCT and TAC are affected by process concurrence relationship (PCR). EPCRCT describes the effect of upstream information on the CTr of current task. When all the relevant upstream tasks are completed, EPCRCT is 1 which means that upstream tasks don't affect CTr any more. TAC represents the percentage of task which is not constrained by upstream tasks and has not been completed. The policy of a company also affects TAC. Some companies would like to start tasks as early as possible to reduce project cycle time, others would like to start tasks when most of the required information is available in order to reduce rework and costs. For simplicity, we assume that the policy of a company remains consistent in the short term.

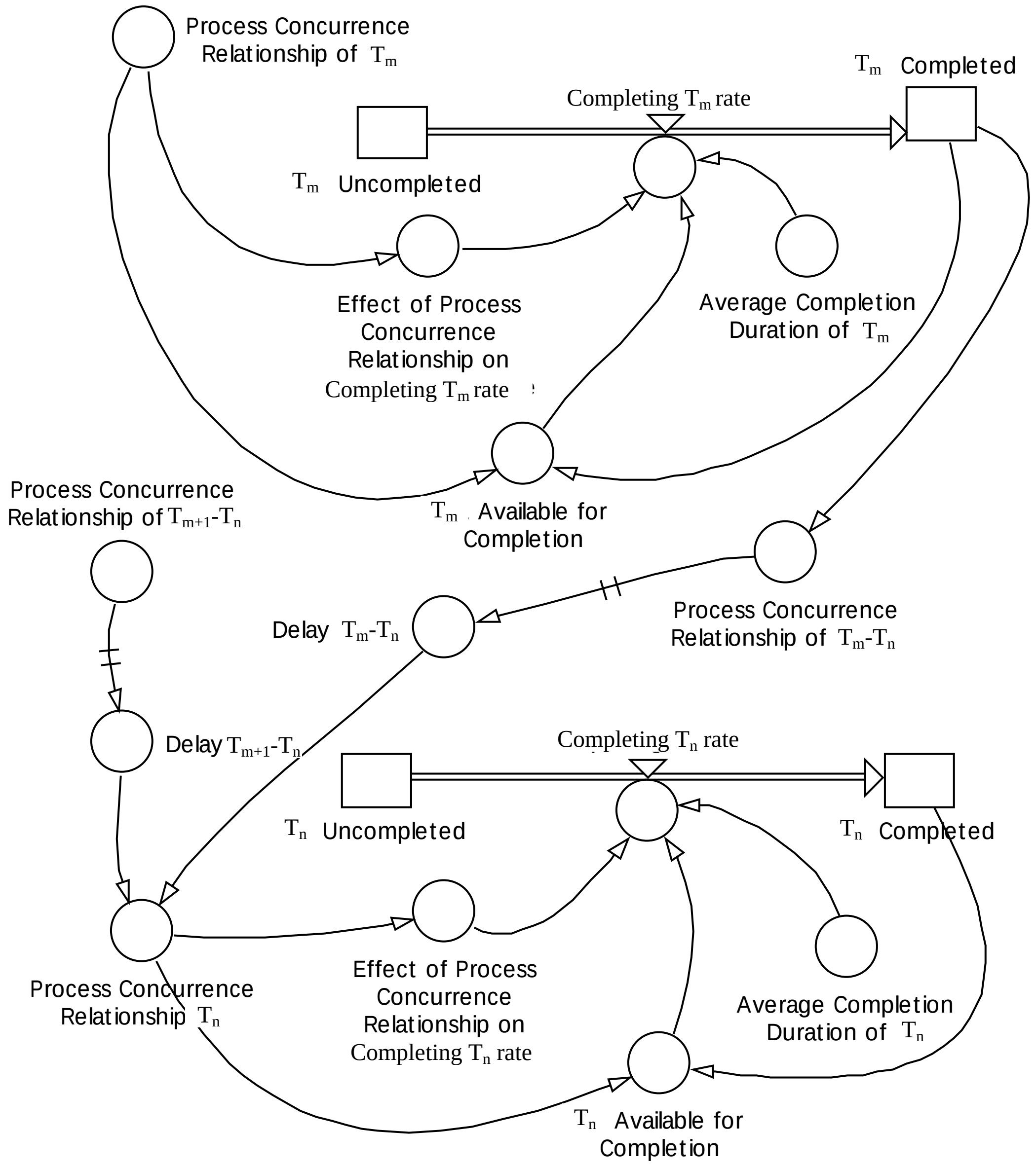
Process Concurrence Relationship of T_m
T_m Completed
Completing T_m rate
T_m Uncompleted
Effect of Process Concurrence Relationship on Completing T_m rate
Average Completion Duration of T_m
Process Concurrence Relationship of T_{m+1}-T_n
T_m Available for Completion
Delay T_m-T_n
Process Concurrence Relationship of T_m-T_n
Delay T_{m+1}-T_n
Completing T_n rate
T_n Uncompleted
T_n Completed
Process Concurrence Relationship T_n
Effect of Process Concurrence Relationship on Completing T_n rate
Average Completion Duration of T_n
T_n Available for Completion

Figure 3: Initial development processes

Process Concurrence Relationships

Process concurrence relationship (PCR) describes the relationship between the percentage of upstream task finished and the percentage of downstream task available. Sometimes one downstream task has several upstream tasks. All the upstream tasks will constraint certain proportion of the downstream task available. In our model PCR of a task (PCR_n) is the tightest constraint set by the PCRs between the focal task (denoted with subscript n) and the upstream development tasks (denoted with subscript m where $m \in \{1,2...,n-1\}$) (PCR_{mn}). If upstream phases do not constrain downstream focal phase n, PCR is 1. For unconstrained tasks PCR and EPCRCT are always 1. This means that all the information needed for unconstrained tasks is available. In distributed product development processes, information delay (ID) often happens because of cooperation problems between teams or organizations. We treat ID as an important part of our model.

The relationships of different tasks can be sequential, concurrent or partially concurrent. For a sequential development process, all information needed for downstream tasks is generated at the time when all (i.e. 100 percent) of the upstream tasks are completed. For concurrent development process, information is formed gradually as each task is being completed. Then information is transferred to downstream tasks. The downstream task will start when there is enough information from its upstream tasks. The development progress can be constrained by inter-task relationships. Consider the case of the mobile phone development as an example. In the design verification test phase, co-developers and standard part suppliers can only design and produce the components after the prototype has been tested. The assembly of these components is constrained by the availability of the mechanical and electrical components, such as the keypad, printed circuit board etc. If there is any quality problem found, the phones must be repaired or reproduced. For any given technology, a certain amount of time is required for each of these tasks. Production, inspection and rectification cannot be executed at the same time in this example. For most projects, although some tasks can be done simultaneously, the other tasks have to be done sequentially, thus constraining the cycle time of the development project.

PCRs of a project describe the interdependency of the tasks, which constraints the development speed of the project. Most previously published system dynamics models of projects have assumed that all the tasks are available for completion, or these constraints have been incorporated into other project features (e.g., Richardson and Pugh 1981; Abdel-Hamid 1984). The assumption that all tasks are available implies that the project can be done immediately when there are enough resources. New product development research (e.g., Clark and Fujimoto 1991; Wheelwright and Clark 1992; Ford and Sterman 1998) shows that the development process is one of the most important aspects that constrain the availability of tasks and the new product development cycle time.

PCRs capture the degree of concurrence of tasks and the changes in the degree of concurrence as the task progresses. As shown in Figures 4(a) and 4(b), when T_m (upstream task) available for T_n (downstream task) is below a certain level, PCR_{mn} (process concurrence relationship of task m and task n) remains zero. It will increase as more T_m becomes available. The highest point is reached after the proportion of T_m which constrains T_n is available. PCR can be applied to any task which is constrained by upstream tasks. The percentage of task available due to upstream constraint is a function of the percentage of available upstream tasks in the development processes. A variety of function forms are possible, such as linear or non-linear relationships, as long as the differential coefficient of the function remains positive. In general strong concurrent relationships are described by curves near the vertical axis of the relationship graph and weak concurrent relationships are described by curves near the horizontal axis of the graph.

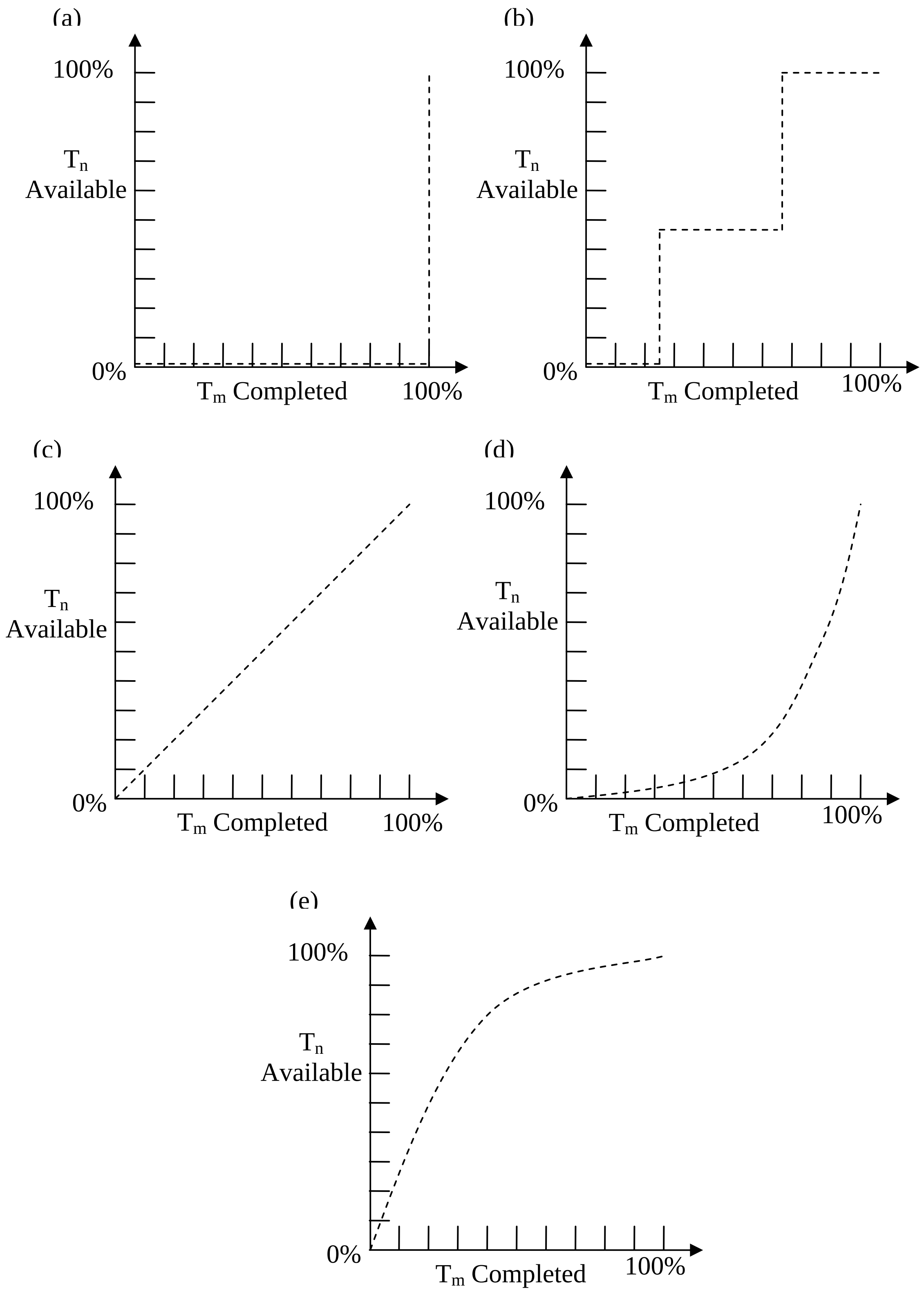

Figure 4: Examples of process concurrence relationships of T_m-T_n : (a) sequential relationship; (b) discrete concurrence relationship; (c) "lockstep" concurrence relationship; (d) low concurrence relationship; (e) strong concurrence relationship

Figure 4 illustrates several possible PCRs. Figure 4(a) shows the PCR which also can be illustrated by the Critical Path Method and PERT. The other figures show PCR that cannot be described with the Critical Path method and PERT. Based on simulation, our model shows the relationships at a high level of details by altering the shape of the function curve. For instance, Figure 4(b) shows that the downstream task can only start until 25% of the upstream task is available. The availability of the downstream task increases discretely according to the percentage of upstream task information available. Figure 4(c) shows that the availability of the downstream task increases continuously as the percentage of available upstream task information increases. Figures 4(d) and 4(e) show the relationships in less concurrent processes and more concurrent processes, respectively.

Rework Processes

We have discussed the simulation of the development process if there is no quality problem and rework. What should the development process be if there are quality problems? According to our assumption that the task is either correct or wrong, if quality problems are found, the relevant tasks have to be redone. For example, the feed back flow from T_n to T_m in Figure 5 means that we find quality problems at state T_n Completed. In order to solve these problems, T_m and all the downstream tasks (T_{m+1} and T_n) have to be redone. The model for the rework processes are based on quality-related information flows and task flows, and explicitly include the discovery of the quality problems, finding the responsible tasks, and rectification of the quality problems in the development processes.

Feedback Information Flows

The development activities may not be perfect, causing some quality problems to fail to be corrected. Some tasks that have quality problems can be mistakenly considered to be finished such that wrong information gets transferred to downstream tasks. The rework information flows arise when these problems are found at downstream task.

As shown in Figure 5, the decision point (DP) describes the time when decision of rework (DOR) should be made. DOR represents progress reviews that determine whether to proceed to the next task or go back to upstream tasks. If rework happens, PCR of the current task and the downstream tasks will become zero and development activity of the downstream tasks will be stopped. The alternatives at the DOR are tagged with probability of rework, which is determined according to the total task completed (TTC) by that time. This means that rework probabilities may be changed as a function of the number of times that the task has been repeated. Improved understanding decreases the probability of iteration. This learning effect, which captures the influence of knowledge accumulation, occurs through the dynamic updating of probability of iteration according to TTC. Cooperation of the teams involved also affects the probability of rework, because information exchanged by teams highlights problems before they turn into rework. Quality information delay (QID) describes the time needed to start rework and it is affected by the coordination efficiency between the team which find the quality problem and the team which is responsible for the problem. All relevant completed tasks will return to uncompleted tasks, as rework information transferred from QID to task revision (TR).

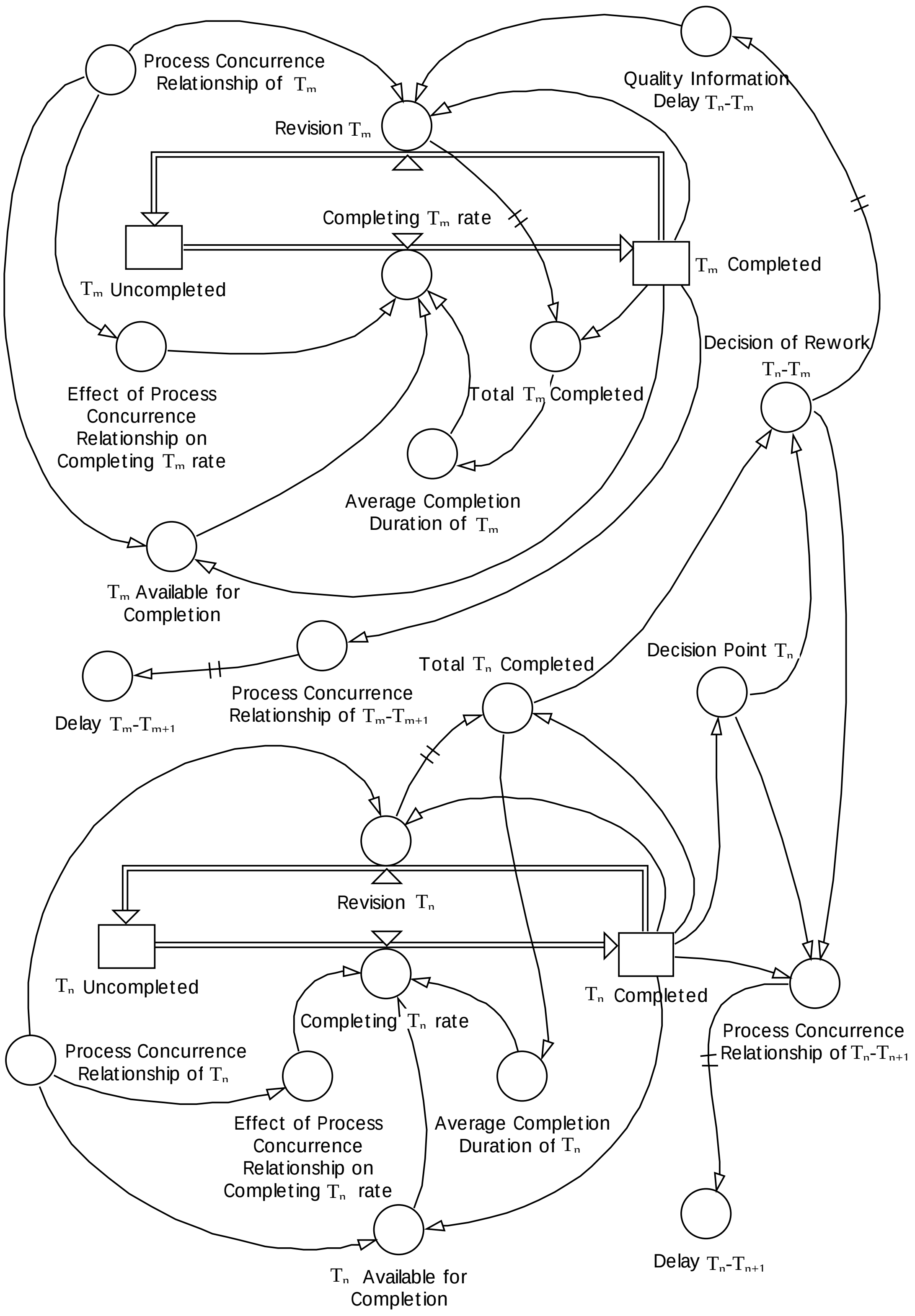
Process Concurrence Relationship of T_m
Quality Information Delay T_n-T_m
Revision T_m
Completing T_m rate
T_m Uncompleted
T_m Completed
Decision of Rework T_n-T_m
Effect of Process Concurrence Relationship on Completing T_m rate
Total T_m Completed
Average Completion Duration of T_m
T_m Available for Completion
Delay T_m-T_{m+1}
Process Concurrence Relationship of T_m-T_{m+1}
Total T_n Completed
Decision Point T_n
Revision T_n
T_n Uncompleted
T_n Completed
Completing T_n rate
Process Concurrence Relationship of T_n
Effect of Process Concurrence Relationship on Completing T_n rate
Average Completion Duration of T_n
Process Concurrence Relationship of T_n-T_{n+1}
T_n Available for Completion
Delay T_n-T_{n+1}

Figure 5: Rework Processes

Adjustment of Task Flows

Learning may take place during the iteration. Improved understanding may increase the CTr and cooperating speed, so that the ACD should be a function of TTC, which is the sum of TC and the integration of task flow through the TR.

The rework of upstream tasks will affect rework of all the downstream tasks. This means PCR can affect TR. For example, if rework happens for T_m the TC_m stock and $PCR_{m(m+1)}$ (process concurrence relationship of task m and task m+1) will become zero. Then the change of $PCR_{m(m+1)}$ will cause rework of T_{m+1}.

Case Background

The company where the case study was conducted is a design company which mainly develops mobile phones. This company operates in a business-to-business market, meaning that its customers are other companies, not end users. It develops mobile phones according to market and technology trends, and then sells the design to customers who will in turn sell the products to end users. The market for these products has strong time and cost pressures. In terms of product architecture, the electronic architecture is designed on the platform by a large international company, while the mechanical architecture and software are developed in-house.

The products analyzed in this case study were developed based on customer requirements and the customer was involved from the beginning of the product development process. Other co-developers in this project were tooling companies, which developed and manufactured non-standard parts; original equipment manufacture, which assembled the mobile phones and tested their performance; and electronics suppliers. The development processes of these products comprise six phases:

1) *Concept Development*: Based on the requirements provided by the customer, the design company defined the main features with the customer, and developed the industrial design plan, mechanical design plan, and electronic design plan.
2) *Industrial Design (ID):* This stage has several sub-processes: initial design, 3D model, rendering, checking, dummy sample and ID confirm. The customer attended the checking process to identify quality problems related to the features and appearance of the mobile phone.
3) *Detail Design (DD):* At this stage, the mechanical design, electronic design and, software design started. Working samples were completed. Then a team from the design company checked the mechanical and electronic performance of the working sample with customer.
4) *Engineering Verification Test (EVT):* Dozens of mobile phones were produced at this stage. The mechanical prototypes were used in this stage. These products were assembled in the design company in order to identify and solve problems related to the assemble process. The mechanical and electronic performances were tested.
5) *Pilot Run (PR):* Half-way through the DD process, the project manager began to prepare for mould fabrication. According to the quality information from EVT, the design and mould were modified. The mobile phones were produced by a contract-manufacturing company. Quality engineers in the manufacturing company tested product quality and provided a report to the design company. Finally, quality problems of design, mould and production process were solved at this stage.
6) *Pilot-line Production (PP):* About one thousand products were produced at this stage to

make sure that problems related to mass production were identified and addressed before the actual mass production began.

Figure 6 shows the development process of the design company. From Figure 6, a number of conclusions could be drawn. Firstly a lot of tasks are done outside the company, so the cooperation among the companies may affect project cycle time; secondly most of the phases are done by several teams belong to different companies, phase based modeling can't explicitly model the information flows in the product development process.

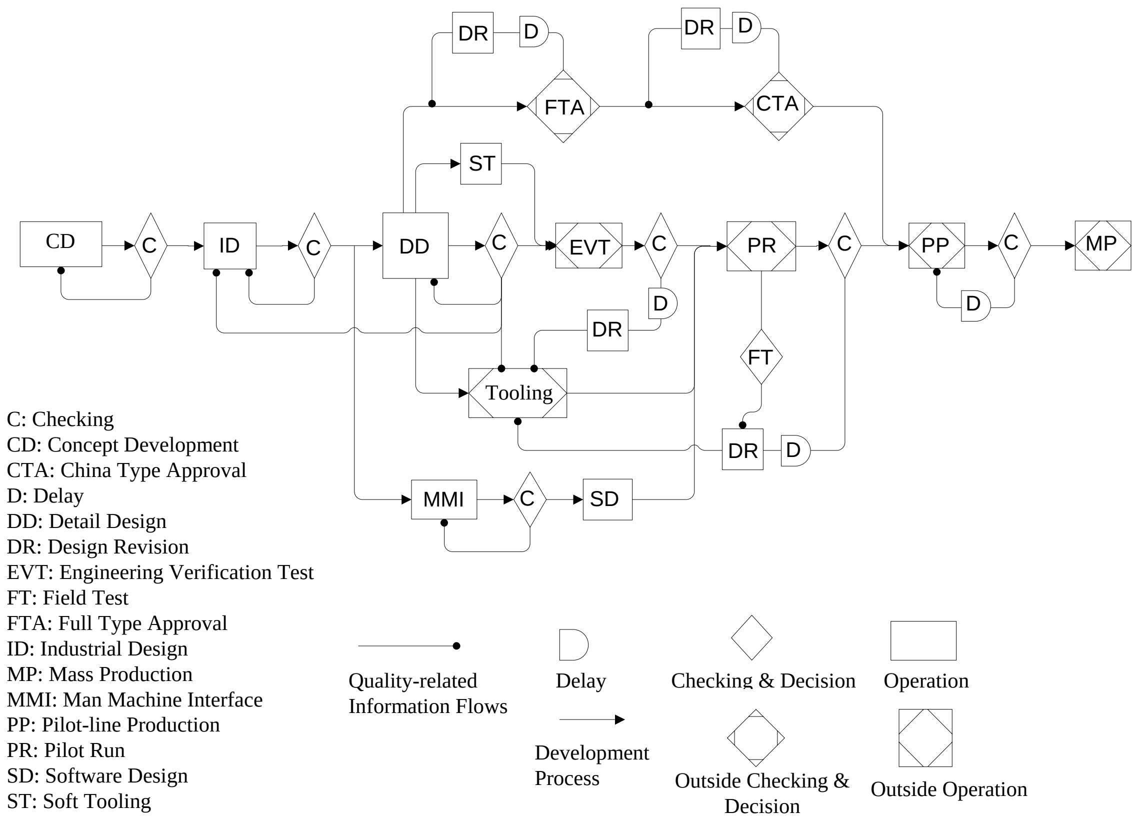

Figure 6: Product development process of the company

Results

The model for the case studied was built based on interviews with the project managers, mechanical engineers and quality engineers, and detailed historical data of four projects done from April 2003 to May 2004. The types and sources of data are listed in Table 2. All projects are derivative product development projects, where the product platform has been established. The data are confidential and are not replicated here exactly. For the purpose of illustration, we multiply all the original task duration by a confidential scale. We formally model the development process from the start point of Industrial Design to the start point of Mass production. Although concept development is very important for NPD processes, we didn't model this part because of its high uncertainty nature.

Table 2 Project cycle-time of the projects done in the focal company

Sources / Types	Interview	Documents
General information about the company	General manager; vice general manager (market)	Project management handbook
General information about the development projects in the company	Vice general manager (Development); technical director	Project management handbook; quality control procedure; standard operation procedure
Detailed data of the 4 projects	Project managers;mechanical engineers;quality engineers	Project progress reports; quality reports

In order to study the usability of the model, we compared the cycle times of ten derivative projects completed in 2004 (there are 21 projects completed in 2004 of the company studied, but some of these are relatively novel, and some of these are not full projects) and the simulation results of our model. We list the data from real projects and simulation runs in Table 3 and Table 4:

Table 4-3 Project cycle-time (working days) of the projects done in the focal company

Project	Cycle-time (working days)	Project	Cycle-time (working days)
1	198	6	184
2	159	7	163
3	163	8	164
4	152	9	171
5	158	10	165

Table 4-4 Project cycle-time (working days) of 32 runs of the model

175	175	154	196
168	168	172	150
185	173	167	174
169	162	173	193
168	151	174	175
166	174	170	168
193	176	168	160
183	172	173	172

We use independent t-test to study the difference of the cycle-time of the two populations. Because the projects 1-4 are used to build our model, the data generated by the model should be closely related to these projects. The other 6 independent projects are used to test our model. We assume both populations are normal and have uniform variance. The sample mean and sample variance of the independent projects and simulated projects are:

$\overline{x_{r2}} = 167.5 \quad s_{r2}^2 = 84.73$

$\overline{x_s} = 171.78 \quad s_s^2 = 111.34$

Pooled variance $s_{p2}^2 = 107.64$

Now test the hypothesis that the difference between two means μ_s and μ_{r2} is zero ($\alpha = 0.05$)

$H_0 : \mu_s - \mu_{r2} = 0$

$H_1 : \mu_s - \mu_{r2} \neq 0$

$$t = \frac{(\overline{x_s} - \overline{x_{r2}}) - 0}{\sqrt{s_{p2}^2 (\frac{1}{n_s} + \frac{1}{n_{r2}})}} = 0.856\,,\ d.f. = 36$$

Critical regions: t<-1.689 and t>1.689

Conclusion: Results from the two populations are not statically different.

F-test is carried out to check the assumption of uniform variance for the two populations:

$$F = \frac{s_s^2}{s_r^2} = 1.314, d.f. = 31,5$$

From the F-table, $F_{0.95}(31,5) = 4.50$. Hence there is insufficient evidence to reject the assumption that the two populations have the same variance.

In order to study the influence of information flows among development teams on project performance, we interviewed the project managers, mechanical engineers and quality engineers related to the projects 1-4. We asked them what the task duration, development rate, rework rate, and iteration probability would be, if all the information needed from other teams available at the right time (The data obtained from mobile phone companies having high level of quality-related information flows are used as reference). The data related to Current Level come from the projects 1-4 which have serious information flow problems among design team, tooling team, manufacturing team and testing team. Then we use these data as inputs of the model. We run the model to study the inference of information flows on project performance. Table 5 shows the simulation results when the information flows are improved. Ten runs were carried out for each of the results. The last column shows the time needed to finish a project. The impacts of the information flows related to the customers and standard part suppliers are not discussed here, since there is no serious information flow problems exist.

Table 5 Simulation Results of Project Cycle Time

Level of Quality-related Information Flows between DC and OEM	Level of Quality-related Information Flows between DC and TC	Cycle Time	Percentage of Improvement
Current Level	Ideal Level	139.2	18.97%
Ideal Level	Current Level	141.6	17.57%
Ideal Level	Ideal Level	116.1	32.41%

References

Abdel-Hamid, T. K. 1984. The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective. Doctoral thesis, MIT, Cambridge, MA.

Bhuiyan, N., D. Gerwin and V. Thomson. 2004. Simulation of the New Product Development Process for Performance Improvement. *Management Science* 50(12), 1690-1703.

Brombacher, A. C. 1999. Maturity index on reliability: covering non-technical aspects of IEC 61508. *reliability certification. Reliability Engineering & System Safety* 66, 109–120.

Clark, K. B. and T. Fujimoto. 1991. *Product Development Performance Strategy, Organization and Management in the World Auto Industry*. Boston, MA: Harvard Business School Press.

Doz, Y. and G. Hamel. 1998. *Alliance Advantage: the art of Creating Value through Partnering*. Harvard Business School Press, Boston, MA.

Engardio, P. and B. Einhorn. 2005. Outsourcing Innovation. BusinessWeek Magazine March 21, 2005.

Eppinger, S. D., D. E. Whitney, R. P. Smith and D. A. Gebala. 1994. A model-based method for organizing tasks in product development. *Research in Engineering* Design 6(1), 1-13.

Ford, D. N. and J. D. Sterman. 1998. Dynamic modeling of product development processes. *System Dynamics Review* 14(1), pp31-68.

Lu, Y. 2002. Analysing Reliability Problems in Concurrent Fast Product Development Processes. Ph.D. thesis, Eindhoven University of Technology, Eindhoven.

Magee, J. F. 1992.Strategic alliances: Overcoming barriers to success. *Chief Executive* 81(11/12), 56-60.

McDonough III, E. F., K. B. Kahn, and G. Barczak. 2001. An investigation of the use of global, virtual, and collocated new product development teams. *Journal of Product Innovation Management* 18(2), 110-120.

Minahan, T. 1998. Platform teams pair with suppliers to drive Chrysler to better designs. *Design News*, 53(10), s3-s7

Proctor, P. 1999. Boeing hones new 550-seat transport design. *Aviation Week & Space Technology* 150(17), 39.

Richardson, G. P. and A. L. Pugh III. 1981. *Introduction to System Dynamics Modeling with Dynamo*. Cambridge, MA: MIT Press.

Sander, P. C. and A. C. Brombacher. 1999. MIR: The use of reliability information flows as a maturity index for quality management. *Quality and Reliability Engineering International* 15, 439-447.

Siemieniuch C. E. and M. Sinclair. 1999. Real-time collaboration in design engineering: an expensive fantasy or affordable reality. *Behaviour & Information Technology* 18(5), 361–371.

Smith, R. P. and S. D. Eppinger. 1997. A predictive model of sequential iteration in engineering design. *Management Science* 43(8), 1104-1120.

Sosa, M. E., S. D. Eppinger, M. Pich, D. G. McKendrick, and S. K. Stout. 2002. Factors that influence technical communication in distributed product development: An empirical study in the telecommunications industry. *IEEE Transactions on Engineering Management* 49(1), 45-58.

Steward, D. 1981. The design structure matrix: A method for managing the design of complex systems. *IEEE Trans. Engineering Management*, 28(3), 71-74.

Wheelwright, S. C. and K. B. Clark. 1992. *Revolutionizing Product Development, Quantum Leaps in Speed, Efficiency, and Quality*. New York: The Free Press.

Williams, T., C. Eden, F. Ackerman and A. Tait. 1995. The effects of design changes and delays on project cost. *The Journal of the Operational Research Society* 46, 809-818.

Zhao, Y. 2003. The Trend toward Outsourcing in New Product Development: Case Studies in Six Firms. International Journal of Innovation Management 7(1), 51–66.